UNITED STATES PATENT APPLICATION

for

A CACHE MECHANISM

Inventors:

Ryan Rakvic
Youfeng Wu
Bryan P. Black
John P. Shen

Prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard
Los Angeles, CA 90025-1026
(303) 740-1980

File No.: 042390.P18230

A Cache Mechanism

FIELD OF THE INVENTION

[0001] The present invention relates to integrated circuits; more particularly, the present invention relates to interfacing integrated circuits.

BACKGROUND

[0002] Almost all computer systems currently in existence implement a cache memory system. A cache memory is typically a random access memory (RAM) that a computer microprocessor accesses more quickly than it can access main memory. As the microprocessor processes data, it looks first in the cache, and if it finds the data in the cache, the microprocessor does not have to do the more time-consuming reading of data from larger main memory.

[0003] Typically, computer systems implement a conventional cache organization that implements caches L1 through L2 or L3. When processing data, the microprocessor first looks in the L1 cache for the data. If the data is not found in the L1 cache, the L2 cache is searched for the data. Finally the L3 cache is searched if the data is not found in the L2 cache.

[0004] A problem with the conventional cache organization is that data associated with various instructions are more critical than others, and thus need to be processed faster (e.g., 1 latency cycle). However, conventional cache operation does not account for the fast retrieval of such critical data from the cache system. For example, both critical data and non-critical data (e.g., data requiring up to 4 latency cycles) may be stored in the L1 cache. Storing non-critical data at the fast L1 cache is inefficient because it reduces the number of critical loads data that may be stored in the L1 cache. Thus, the larger, slower L2 or L3 caches must be searched to retrieve critical data that will are not stored within the L1 cache, exceeding the 1 latency cycle.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The present invention will be understood more fully from the detailed description given below and from the accompanying drawings of various embodiments of the invention. The drawings, however, should not be taken to limit the invention to the specific embodiments, but are for explanation and understanding only.

[0006] **Figure 1** illustrates one embodiment of a computer system;

[0007] **Figure 2** is a flow diagram for one embodiment of load classification; and

[0008] **Figure 3** illustrates one embodiment of load placement within cache memories.

DETAILED DESCRIPTION

[0009] A cache mechanism is described. According to one embodiment, the cache mechanism features a modified cache hierarchy that includes a vital cachelet that does not cache data for semi-vital and non-vital loads. This enables the vital cachelet to be more efficiently utilized by a smaller set of loads (e.g., vital loads). Similarly, a semi-vital cachelet does not cache data for non-vital loads. In a further embodiment, loads are directly assigned to the various cachelets, which are accessed in parallel by different loads. Further, load assignment is performed statically, and load/store units can be directly attached to the cachelets.

[0010] Reference in the specification to "one embodiment" or "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the invention. The appearances of the phrase "in one embodiment" in various places in the specification are not necessarily all referring to the same embodiment.

[0011] In the following description, numerous details are set forth. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention

[0012] **Figure 1** is a block diagram of one embodiment of a computer system 100. Computer system 100 includes a central processing unit (CPU) 102 coupled to bus 105. In one embodiment, CPU 102 is a processor in the Pentium® family of processors including the Pentium® II processor family, Pentium® III processors, and Pentium®-IV processors available from Intel Corporation of Santa Clara, California. Alternatively, other CPUs may be used.

[0013] A chipset 107 is also coupled to bus 105. Chipset 107 includes a memory control hub (MCH) 110. In one embodiment, MCH 110 is coupled to an input/output control hub (ICH) 140 via a hub interface. ICH 140 provides an interface to input/output (I/O) devices within computer system 100. For instance, ICH 140 may be coupled to a Peripheral Component Interconnect bus adhering to a Specification Revision 2.1 bus developed by the PCI Special Interest Group of Portland, Oregon.

[0014] In one embodiment, MCH 110 includes a memory controller 112 that is coupled to a main system memory 115. Main system memory 115 stores data and sequences of instructions and code represented by data signals that may be executed by CPU 102 or any other device included in system 100. In one embodiment, main system memory 115 includes dynamic random access memory (DRAM); however, main system memory 115 may be implemented using other memory types.

[0015] According to one embodiment, cache memories 152, 154 and 156 are coupled to CPU 102. In a further embodiment, caches 152, 154 and 156 implement a novel cache organization that provide load data to CPU 102 as the data is needed by dependent instructions. Thus, the number of tardy data arrivals at CPU 102 is reduced and caches are more efficiently utilized, resulting in improved CPU performance.

[0016] In one embodiment, loads are classified by vitality, and are directly assigned to caches 152, 154 and 156, which are small and fast caches, referred to as cachelets. In a further embodiment, the cachelets are designed with different latencies to better track load behavior. Load vitality is defined based on the distance between a load and its use (e.g., load-use distance).

[0017] According to one embodiment, loads are classified into three classes: vital (use immediately), semi-vital (use in 2 to 3 cycles), and non-vital (no use for at least 4 cycles). These loads are then assigned to appropriately timed cachelets: cache 152 (e.g., vital cachelet (single cycle)), cache 154 (e.g., semi-vital cachelet (2 cycles)), and cache 156 (e.g., the non-vital cache (4 cycles)). Although described herein with respect to three cachelets, one of ordinary skill in the art will appreciate that other quantities of cachelets (e.g., 4, 5, 6, etc.) may be implemented.

[0018] In one embodiment, cache 152 is a very fast (e.g., 1 cycle latency) and very small (e.g., 256B) cache that stores data only for vital loads. Cache 152 is placed in parallel with cache 154, which is a fast (e.g., 2 cycle latency) and small (e.g., 1KB) cache. Load data that is needed immediately is assigned to the single cycle cache 152. Loads that are semi-vital are assigned directly to cache 154. In one embodiment, cache 152 and cache 154 together operate at the same level in the cache hierarchy. All non-vital loads are assigned to cache 156 that has a 4-cycle latency. If a vital load misses cache 152, it then accesses cache 156.

[0019] According to one embodiment, allocation of data is carried out based on load classification. Thus, vital loads update cache 152 and cache 156, while semi-vital loads update the cache 154 and cache 156. Non-vital loads allocate data in cache 156. Note that a coherence issue may occur between cache 152 and cache 154 because both may have the same data simultaneously.

[0020] To accommodate this, stores are broadcasted to both cache 152 and 154, as well as cache 156. Such store bandwidth is not a performance limiter. This on-demand allocation and replication allows the caches to be more efficiently utilized. The inclusion principle that is evident in a conventional cache organization is not enforced between caches 152 and 154. Thus, there is more effective cache space with in the disclosed system when compared to a conventional cache organization.

[0021] To implement the disclosed cache mechanism, load classification is to be performed prior to cache access. In one embodiment, a compiler performs load classification. However in other embodiments, load classification may be performed using other software, or hardware mechanisms. The compiler knows the distance between most loads and their uses. Consequently, the compiler does an effective job at vitality classification.

[0022] **Figure 2** is a flow diagram for one embodiment of load classification. At processing block 210, vital loads are identified. In one embodiment, vital loads are identified by filtering out all loads with a dependence distance greater than 1 cycle. Within this process, there is a pre-scheduling phase and a post-scheduling phase to maximize non-vital classification. Subsequently, profiling is performed to identify the remaining loads that miss frequently in cache 152 and lines that are not reused.

[0023] At processing block 220, semi-vital loads are identified. In one embodiment, information from the vital load identification and profiling are used to determine the semi-vital loads. Profiling is again performed to identify the loads that miss frequently in cache 154 and lines that are not reused. At processing block 230, non-vital loads are identified. In one embodiment, the profile information from vital load and semi-vital identification is used to determine the loads that frequently miss in cache 154 and with cache lines that are not reused.

[0024] **Figure 3** illustrates one embodiment of load placement within cache memories 152, 154 and 156. As shown in **Figure 3**, vital loads are stored in cache 152, while semi-vital and non-vital loads are stored at caches 154 and 156, respectively.

[0025] The above-described cache mechanism features a reorganized and optimized data cache hierarchy that takes advantage of the fact that not every load data is needed immediately. Thus, assignment of a load to the cache hierarchy is not done with the traditional top-down approach. Instead the assignment of a load and the allocation of its data is performed by a vitality classifier. Non-vital loads are assigned to the slowest cache, and the more vital load(s) are assigned to the fast and small cache(s). The fast and small cache(s) only allocate data for the more vital loads, and provide a high hit rate for these loads.

[0026] The cache mechanism benefits from the direct assignment of loads to the various cachelets, and provides more bandwidth to the CPU core since the cachelets are accessed in parallel by different loads. Another benefit of the cache mechanism is the cache access latency. Load assignment is performed statically, and load/store units can be directly attached to the cachelets.

[0027] Whereas many alterations and modifications of the present invention will no doubt become apparent to a person of ordinary skill in the art after having read the foregoing description, it is to be understood that any particular embodiment shown and described by way of illustration is in no way intended to be considered limiting. Therefore, references to details of various embodiments are not intended to limit the scope of the claims which in themselves recite only those features regarded as the invention.